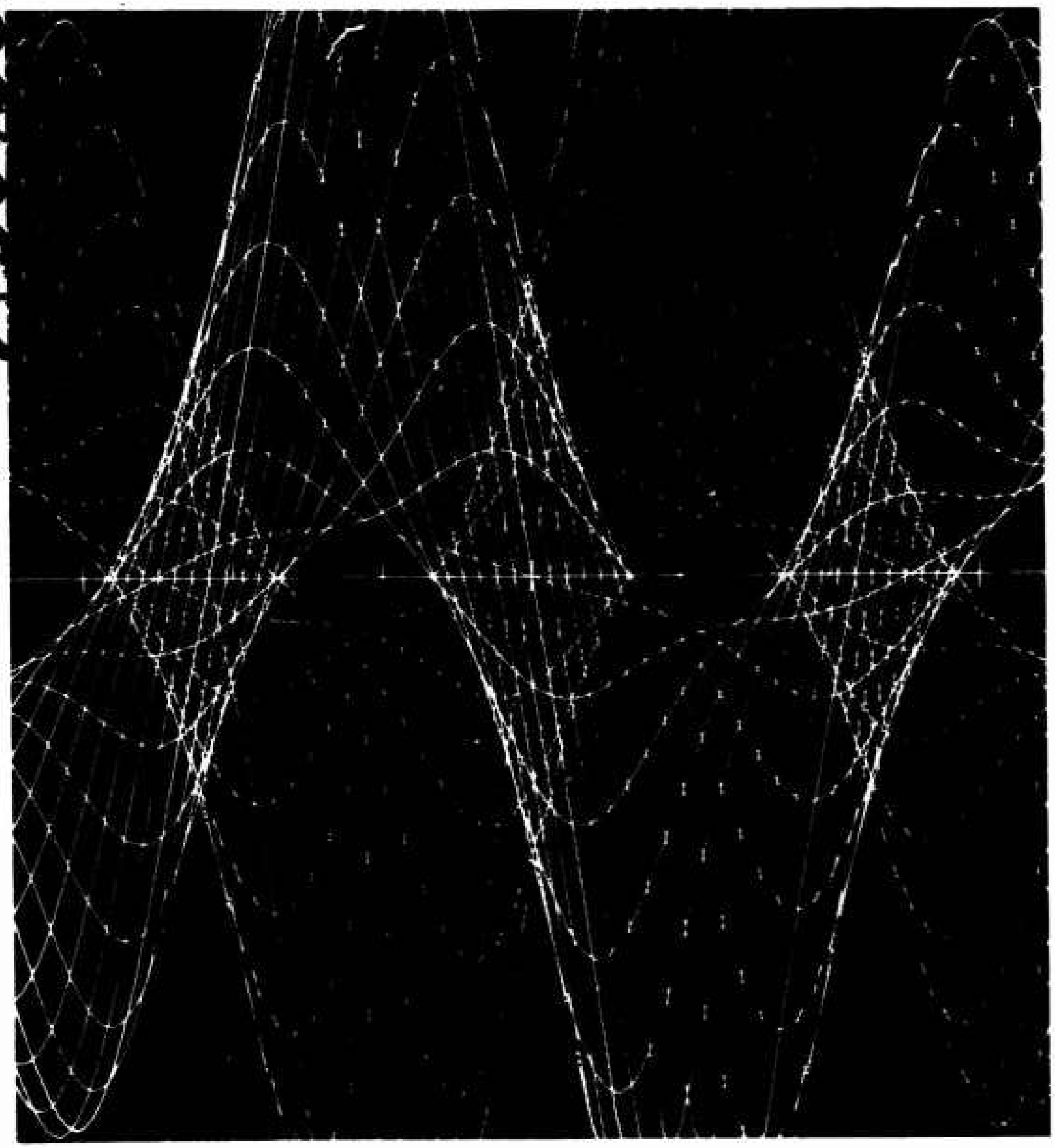
AD696645



RC 2285

ON THE NUMBER OF MULTIPLICATIONS NECESSARY TO COMPUTE CERTAIN FUNCTIONS

Shmuel Winograd

November 15, 1968

IBM RESEARCH

39

The cover is designed from an example of an interaction between computers and the physical and mathematical sciences. The design depicts ion trajectories in a type of mass spectrometer used for chemical analysis of residual gases in ultra high vacuum systems. These ion trajectories represent solutions of Mathieu's differential equation. They are generated by numerical integration of the equation using a high speed computer, and are plotted automatically from an output tape as part of the Research Center computing service.

Printed in the U.S.A.

ON THE NUMBER OF MULTIPLICATIONS NECESSARY TO COMPUTE CERTAIN FUNCTIONS *

Shmuel Winograd
Research Division
Yorktown Heights, New York

ABSTRACT: The number of multiplications and divisions required in certain computations is investigated. In particular, results of Pan and Motzkin, about polynomial evaluation as well as similar resu about the product of a matrix by vector, are obtained. As an application of the results on the product of a matrix by vector, a new algorithm for matrix multiplication, which requires about $\frac{1}{2}n^3$ multiplications, is obtained.

* The results reported in this paper were obtained in the course of research jointly sponsored by the Office of Naval Research and IBM.

RC 2285 (# 11230)
November 15, 1968
Combinatorial mathematics

1. INTRODUCTION

The number of multiplications required to evaluate a polynomial was first investigated by Ostrowski [1]. He showed that in order to evaluate a polynomial $P_n(x)$ of degree n at least n multiplications are required, for $n = 1, 2, 3, 4$. The results were extended by Pan [2], who proved that n multiplications are necessary to evaluate $P_n(x)$ for any n; moreover, if divisions are allowed, then at least n multiplications/divisions are necessary to evaluate $P_n(x)$.

Motzkin [3] introduced the notion of preconditioning of the coefficients. Motzkin showed that if in the course of computing $P_n(x) = \sum_{i=0}^{n} a_i x^i$ operations which depend only the a_i's are not counted, then only about $n/2$ multiplications are necessary to evaluate $P_n(x)$. The obvious application of this result is when the same polynomial $P_n(x)$ has to be evaluated at many different points x.

In this paper we will consider the question of the number of multiplications/divisions (which we shall abbreviate as m/d) necessary to compute several functions. Pan's result about polynomial evaluation as well as the number of m/d required to multiply a matrix by a vector will follow as corollaries of Theorem 1. Theorem 2 deals with the problem of preconditioning, and Motzkin's result is obtained as a special case of this theorem. As an application of the result on preconditioning for the product of a matrix by a vector, we obtain a new algorithm for the product of two $n \times n$ matrices which grows as $\frac{1}{2}n^3$. In Theorem 3 we show that if the number of multiplications required to multiply two $n \times n$ matrices increases as n^3, then at least $\frac{1}{2}n^3$ multiplications are required.

The statement of Theorem 1 was previously reported in [4], and the new algorithm for matrix multiplication in [5].

2. FORMULATION OF THE PROBLEM

In trying to determine the number of m/d required to evaluate the polynomial $P_n(x) = \sum_{i=0}^{n} a_i x^i$, we are interested in the "worst case" polynomial. If the polynomial happened to be, for example, $\sum_{i=0}^{n} x^i = \frac{x^{n+1}-1}{x-1}$ then it may require fewer m/d for its evaluation than a polynomial whose coefficients are all distinct. Similarly, it is easier to evaluate a polynomial at the points $x = 0$ or $x = 1$ than at other points. Intuitively, it seems that the "worst case" occurs when all the coefficients as well as the point x are transcendental numbers with no algebraic relation between them. If only the operations of addition, subtraction, multiplication, and division are allowed in the course of evaluating $P_n(x) = \sum_{i=0}^{n} a_i x^i$ then all the numbers which result during the evaluation are in the field $Q(a_0, a_1, \ldots, a_n x)$, where Q denotes the field of rational numbers. In case all the a_i's as well as x are transcendental numbers without an algebraic relation, this field is isomorphic to the field $Q(a_0, \ldots, a_n, x)$ of the rationals extended by the $n + 2$ indeterminates $a_0, a_1, \ldots, a_n, x$. Thus the problem of "worst case" evaluation of the polynomial $P_n(x)$ is transformed into the problem of constructing the element $\sum_{i=0}^{n} a_i x^i \in Q(a_0, \ldots, a_n, x)$. Conversely, any algorithm which constructs $\sum_{i=0}^{n} a_i x^i$ can be used to evaluate the polynomial for fixed values a_i and x.

The problem we will treat can therefore be formulated as follows. Let F be a field, and $x_1, \ldots, x_n$ be a set of indeterminates. What is the minimum number of m/d necessary to construct the t elements $\psi_j \in F(x_1, \ldots, x_n)$, $j = 1, 2, \ldots, t$.

The concept of a field as the algebraic structure, elements of which are to be constructed, was chosen since the main interest in this paper is the number of m/d. In the definition of an algorithm to be given below we will define an algorithm over any algebraic structure $\mathcal{A} = (X, \Omega)$. We use X to denote the carrier (underlying set) of the algebra, and $\Omega = \{\omega_i \mid i \in I\}$ to denote the set of the (possibly partial) operations. Thus, if ω_i is an n_i-ary operation, then $\omega_i : Y_i \rightarrow X$ where $Y_i \subseteq X^{n_i}$.

In constructing an element $\psi \in \mathcal{A}$, we have to start with a certain subset $\mathcal{B}$ of $\mathcal{A}$ as given, and

construct other elements of $\mathcal{A}$ from the elements of $\mathcal{B}$ using the operations in Ω. In case the algebraic structure is the field $F(x_1,\dots,x_n)$ it may be natural to take $F \cup \{x_1,\dots,x_n\}$ as $\mathcal{B}$.

<u>Definition 1.</u> An N-step <u>algorithm</u> α <u>over</u> $(\mathcal{A},\mathcal{B})$ is a mapping $\alpha : \{1,2,\dots,N\} \to \mathcal{B} \cup (\bigcup_{i \in I} \{\omega_i\} \times \{1,2,\dots,N\}^{n_i})$ subject to the restriction that if $\alpha(k) = (\omega_i, j_1,\dots,j_{n_i})$, then $j_\ell < i$ $\quad \ell = 1,2,\dots,n_i$. With each algorithm α we associate a (partial) function $e_\alpha : \{1,2,\dots,N\} \to \mathcal{A}$ defined by:

(a) $\quad e_\alpha(k) = \alpha(k)$ if $\alpha(k) \in \mathcal{B}$;

(b) $\quad e_\alpha(k) = \omega_i(e_\alpha(j_1),\dots,e_\alpha(j_{n_i}))$ if
$\alpha(k) = (\omega_i, j_1, j_2,\dots,j_{n_i})$ and $e_\alpha(j_\ell)$
$\ell = 1,2,\dots,n_i$ as well as $\omega_i(e_\alpha(j_1),\dots,e_\alpha(j_{n_i}))$
are defined.

The function $\alpha(i)$ is the sequence of operations which the algorithm executes, while $e_\alpha(i)$ is the sequence of elements which the algorithm computes.

<u>Definition 2.</u> The algorithm α is said to <u>compute</u> $\psi_j \in \mathcal{A}$ $\quad j = 1,2,\dots,t$, if e_α is a total function, and if there exist t integers $i_1, i_2,\dots,i_t$ such that $e_\alpha(i_j) = \psi_j$, $j = 1,2,\dots,t$.

The cardinality of $\alpha^{-1}(\{\omega_i\} \times \{1,2,\dots,N\}^{n_i})$ is the number of times the operation ω_i appears in α. If $\mathcal{A}$ is a field, and if we denote multiplication by ω_1 and division by ω_2, then the number of m/d in α is the cardinality of $\alpha^{-1}(\{\omega_1,\omega_2\} \times \{1,2,\dots,N\}^2)$.

An important concept in using algorithms is that of substitution. We will first define substitution within an algebraic structure $\mathcal{A}$, and then extend it to substitution in algorithms.

<u>Definition 3.</u> Let $\mathcal{A}$ be an algebraic structure, and let $\mathcal{B}$ be a subset of the carrier of $\mathcal{A}$. A function $s : \mathcal{B} \to \mathcal{A}$ is said to be a <u>basis for substitution</u> if there exists a partial function $s' : \mathcal{A} \to \mathcal{A}$ called an extension of s such that:

(a) $s'|_{\mathcal{B}} = s$;

(b) If $s'(a_j)$ is defined for $j = 1, 2, \ldots, n_i$, and $\omega_i(s'(a_1), \ldots, s'(a_{n_i}))$ is defined, then $s'(\omega_i(a_1, \ldots, a_{n_i}))$ is defined and $s'(\omega_i(a_1, \ldots, a_{n_i})) = \omega_i(s'(a_1), \ldots, s'(a_{n_i}))$.

<u>Definition 4.</u> Let s be a basis for substitution. We define the partial function $s^* : \mathcal{A} \to \mathcal{A}$ by:

(a) $s^*(a) = s'(a)$ if for any two extensions s' and s'' of s , $s'(a) = s''(a)$;

(b) $s^*(a)$ is not defined if $s'(a)$ is not defined for any extension s' of s ;

(c) $s^*(a) = a$ otherwise.

It is easily verified that s^* is an extension of s and it is unique.

<u>Example.</u> Let $\mathcal{A} = Q[x_1, \ldots, x_n]$; let $\mathcal{B} = \{x_n\}$, and let $s(x_n) = 1$. Then s' defined by $s'(p(x_1, \ldots, x_n)) = p(x_1, \ldots, x_{n-2}, 0, 1)$ is an extension of s (where $p(x_1, \ldots, x_n)$ denotes a polynomial with rational coefficients in the variables $x_1, \ldots, x_n$), while s^* is in this case $s^*(p(x_1, \ldots, x_n)) = p(x_1, \ldots, x_{n-1}, 1)$.

Example. Let $\mathcal{A} = Q(x_1, \ldots, x_n))$, $\mathcal{B} = \{x_n\}$, $s(x_n) = 1$. Then $s^*(\frac{1}{1-x_n})$ is not defined.

<u>Definition 5.</u> Let α be an N step algorithm over $\mathcal{A}$ and let β be a M step algorithm over $\mathcal{A}$ and let $\alpha(k) \in \mathcal{B}$. The <u>algorithm</u> γ <u>resulting from substituting</u> β <u>for step</u> k <u>in</u> α is $N + M - 1$ step algorithm defined as follows. It will be convenient to describe the domain of γ as $\{1, 2, \ldots, k-1\} \cup \{(k, j) \mid j = 1, 2, \ldots, M\} \cup \{k+1, \ldots, N\}$, and where the ordering is $j_1 > j_2$ if and only if $j_1 > j_2$ as integers, $j_1 > (k, j_2)$ if and only if $j_1 > k$ as integers, and $(k, j_1) > (k, j_2)$ if and only if $j_1 > j_2$ as integers. The algorithm γ is defined by:

(a) $\gamma(k, j) = \beta(j)$;

(b) $\gamma(j) = \alpha(j)$ if $\alpha(j) \in \mathcal{B}$;

(c) $\gamma(j) = (\omega_i, f(j_1), \ldots, f(j_{n_i}))$ if $\alpha(j) = (\omega_i, j_1, \ldots, j_{n_i})$ where $f(j) = j$ if $j \neq k$ and $f(k) = (k, M)$.

It is easily verified that if α and β are algorithms, so is γ.

Definition 5 is readily extendable to substituting in α algorithm β_i at step k_i, if $\alpha(k_i) \in \mathcal{B}$ for all i.

<u>Definition 6.</u> Let $s : \mathcal{B} \to \mathcal{A}$ be the basis for substitution, let α be an $(\mathcal{A},\mathcal{B})$ algorithm, and let β_b be an $(\mathcal{A},\mathcal{B}')$ algorithm computing b, for every $b \in \mathcal{B}$. The <u>algorithm</u> γ <u>resulting from</u> α <u>by the substitution</u> s is the $(\mathcal{A},\mathcal{B}')$ algorithm resulting from substituting for step k in α the algorithm β_b whenever $\alpha(k) = b$.

<u>Remark.</u> Even though e_α and e_{β_b} are total, e_γ need not be total, as the following example shows.

<u>Example.</u> Let $\mathcal{A} = Q(x)$, $\mathcal{B} = Q \cup \{x\}$, and let α be the following 7 step algorithm. (We use ω_1 to denote multiplication, ω_2 division, ω_3 addition, ω_4 subtraction.) $\alpha(1) = 1$, $\alpha(2) = x$, $\alpha(3) = (\omega_1, 2, 2)$, $\alpha(4) = (\omega_1, 3, 3)$, $\alpha(5) = (\omega_4, 4, 1)$, $\alpha(6) = (\omega_4, 2, 1)$, $\alpha(7) = (\omega_2, 5, 6)$.

It is easily verified that Q_α is total and that $e_\alpha(7) = \frac{x^4-1}{x-1} = \sum_{i=0}^{3} x^i$. Let s be defined by $s(q) = q$ for every $q \in Q$ and $s(x) = 1$. For this s, $s^*(\frac{x^4-1}{x-1}) = 4$, while the function e_α is not total since $e_\alpha(7)$ is not defined.

It is clear though that if e_α is total, then γ computes $s^*(a)$ whenever α computes a.

One commonly occurring substitution is that of using the algorithm to evaluate a function at a given point. Let F be a subfield of $\mathbb{C}$, and α be an $(\mathcal{A},\mathcal{B})$ algorithm computing $\psi(x_1,\ldots,x_n) \in F(x_1,\ldots,x_n)$, where $\mathcal{A} = F(x_1,\ldots,x_n)$ and $\mathcal{B} = F \cup \{x_1,\ldots,x_n\}$. Let s be defined by $s(x_i) = x_i^0$. This s is a basis for substitution, and let γ be the resulting algorithm. The set of points $(x_1^0,\ldots,x_n^0)$ for which e_γ is not total is a finite union of algebraic surfaces, and therefore, has a measure 0.

In this paper we investigate the number of m/d necessary to compute certain functions in $F(x_1,\ldots,x_n)$ for some field F. Even if a function is in $F[x_1,\ldots,x_n]$ (and, therefore, can be expressed without the division operation), it is possible to minimize the number of m/d

by using division. For example, it is easily verified that at least 7 multiplications are necessary to compute x^{31}, while using $x^{31} = x^{32}/x$ only 6 m/d are required. Of course, if we use division, we have to count the number of multiplications and divisions since $u \cdot v = 1/(1/u)/v)$.

3. THE NUMBER OF MULTIPLICATIONS/DIVISIONS

This section is devoted to investigating the number of m/d necessary to compute elements of the field $F(x_1,\ldots,x_n)$. In particular, we will consider only functions which are linear in the x_i's ; that is, let Φ be a $t \times n$ matrix whose entries $\Phi_{i,j} \in F$, φ a t-vector whose entries $\varphi_i \in F$, and let $\underline{x}$ denote the (column) vector $(x_1,\ldots,x_n)$. We will investigate the number of m/d required to compute the t elements $\Phi\underline{x} + \varphi$. (This notation was suggested by V. Strassen.) In this section we will assume that all algorithms are over $(\mathcal{A},\mathcal{B})$ where $\mathcal{A} = F(x_1,\ldots,x_n)$ and $\mathcal{B} = F \cup \{x_1,\ldots,x_n\}$.

Let G be a subfield of F. We will assume that the number of elements in G is infinite. There is no loss of generality in this assumption, since if $\Phi\underline{x} + \varphi \in F(x_1,\ldots,x_n)^t$, then it is also in $(F(z)(x_1,\ldots,x_n))^t$, and using $F' = F(z)$ and $G' = G(z)$, we obtain that G' has infinitely many elements. We will use $\Phi_1, \Phi_2,\ldots,\Phi_n$ to denote the columns of Φ.

<u>Theorem 1.</u> Let α be an algorithm computing $\Phi\underline{x} + \varphi \in F(x_1,\ldots,x_n)^t$, and let G be a subfield of F.

If there are u vectors in $\{\Phi_1, \ldots, \Phi_n\}$ such that no nontrivial linear combination of them (over G) is in G^t, then α has at least u m/d. Moreover, α has at least u steps k, such that $\alpha(k)$ is of the form (ω_1, j_1, j_2) such that $e_\alpha(j_1) \notin G$ and $e_\alpha(j_2) \notin G$, or $\alpha(k)$ is of the form (ω_2, j_1, j_2) and $e_\alpha(j_2) \notin G$. (That is, multiplication or division by an element of C is not counted.)

Proof. We will use the phrase "m/d which is counted" to denote multiplications such that $e_\alpha(j_1) \notin G$ and $e_\alpha(j_2) \notin G$ or divisions such that $e_\alpha(j_2) \notin G$. (That is, those m/d which are not excluded by the second half of the theorem.)

By induction on q, it is easily verified that if the first q steps α do not involve an m/d which is counted, then for all $i = 1, 2, \ldots, q$, $e_\alpha(i) = \sum_{j=1}^{n} g_j x_j + f$ for some $g_{j,i} \in G$ $j = 1, 2, \ldots,$ $i = 1, 2, \ldots, q$, and some $f_i \in F$.

We will prove the theorem by induction on u (assuming, as was remarked before, that the number of elements in G is infinite).

If $u = 1$, there exists i_0, j_0 such that $\Phi_{i_0, j_0} \notin G$. If no m/d which is counted appears in α, then (since α computes $\Phi \underline{x} + \varphi$) for some k, $e_\alpha(k) = \sum_{j=1}^{n} \Phi_{i_0, j} x_j + \varphi_{i_0}$. But since α has not m/d which is counted, there exist $g_j \in G$ $j = 1, 2, \ldots, n$ and $f \in F$ such that $e_\alpha(k) = \sum_{j=1}^{n} g_j x_j + f$, and therefore, $g_{j_0} = \Phi_{i_0, j_0}$ contradicting the assumption that $\Phi_{i_0, j_0} \notin G$.

Assume the assertion holds for u (and all t and n), and let α be an algorithm computing $\Phi \underline{x} + \varphi$ (α is assumed to be the algorithm minimizing the number of m/d which are counted), and there are at least $u + 1$ vectors in $\{\Phi_1, \Phi_2, \ldots, \Phi_n\}$ such that no nontrivial linear combination of them (over G) is in G^t. Let k be the smallest integer such that an m/d which is counted occurs at step k; then either $e_\alpha(k) = (\sum_{i=1}^{n} g_i x_i + f) \cdot (\sum_{i=1}^{n} h_i x_i + f')$ or $e_\alpha(k) = (\sum_{i=1}^{n} g_i x_i + f) : (\sum_{i=1}^{n} h_i x_i + f')$ for some $g_i, h_i \in G$ $i = 1, 2, \ldots, n$ and $f, f' \in F$. Furthermore, either one of the h_i's or else one of the g_i's is not 0; otherwise, $e_\alpha(k) \in F$ and no m/d would be required in a minimal algorithm. Assume one of the h_i's is not 0.

(If all the h_i's are 0, then one of the g_i's is not 0 and the proof proceeds in the same manner.) With no loss of generality, we can assume $h_n \neq 0$ and since multiplication by h_n or h_n^{-1} are not counted, we can assume $h_n = 1$. Let $g \in G$ be such that if we substitute $g - f' - \sum_{i=1}^{n-1} h_i x_i$ for x_n, the resulting algorithm α' is such that $e_{\alpha'}$ is total. Such a g exists since there are only finite of substitutions for x_n such that $e_{\alpha'}$ is not total, and G has infinitely many elements. (Only finite number of substitutions for x_n cause $e_{\alpha'}$ not to be total since $e_\alpha(i)$ is a fraction of polynomials in x_n and there are only finite number of steps in the algorithm α.)

Substituting $g - f' - \sum_{i=1}^{n-1} h_i x_i$ for x_n we obtain an algorithm which computes $\Phi'\underline{x}' + \varphi'$ where $\Phi'_j = \Phi_j - h_i \Phi_n \quad j = 1, 2, \ldots, n-1$, $\varphi' = \varphi + (g-f')\Phi_n$, $\underline{x}' = (x_1, \ldots, x_{n-1})^T$. The number of m/d which are counted in α' is at least one fewer than in α since step k is not an m/d which is counted in α', and since the algorithm β computing $g - f' - \sum_{i=1}^{n-1} h_i x_i$ has not m/d which is counted. But there exist at least u vectors in $\{\Phi'_1, \Phi'_2, \ldots, \Phi'_{n-1}\}$ such that no nontrivial linear combination of them (over G) is in G^t and by induction hypothesis α' has at least u m/d which are counted, and therefore, α has at least $u+1$ m/d which are counted.

<u>Corollary 1</u> (Pan). No algorithm for evaluating $\sum_{i=0}^{n} x_i y^i$ requires fewer than n m/d, and therefore, Horner's rule minimizes the number of m/d in computing a polynomial.

<u>Proof</u>. Let $F = G(y)$. Then $\sum_{i=0}^{n} x_i y^i = \Phi\underline{x}$ where Φ is the $1 \times (n+1)$ matrix $\Phi_{1,j} = y^j \quad j = 0, 1, \ldots, n$. Since $(1, y, y^2, \ldots, y^n)$ are linearly independent over G, $\Phi\underline{x}$ satisfy the condition of the theorem for $u = n$.

<u>Corollary 2.</u> Let $X = (x_{i,j})$ be a $p \times q$ matrix, and let $\underline{y} = (y_j)$ be a q column vector. No algorithm for computing $X\underline{y}$ requires fewer than pq m/d, and therefore, the ordinary way for computing $X\underline{y}$ minimizes the number of m/d.

Proof. Let $F = G(y_1, \ldots, y_q)$, and let Φ be the $p \times pq$ matrix where

$$\Phi_{i,j} = \begin{cases} y_k & \text{if } j = iq + k \quad 1 \leq k \leq q \\ 0 & \text{otherwise ;} \end{cases}$$

and let $\underline{x}$ be the column vector $(x_{1,1}, \ldots, x_{1,q}, x_{2,1}, \ldots, x_{2,q}, \ldots, x_{pq})$. Clearly $X\underline{y} = \Phi\underline{x}$ and since no nontrivial linear combination (over G) of the columns of Φ is in G^P, the corollary is obtained by applying Theorem 1.

Remark. The results of the corollaries hold even if we do not count m/d involving only y (in Corollary 1) or $y_1, \ldots, y_q$ (in Corollary 2), since in the statement of the theorem, we allowed forming any element of $G(y)$ (in Corollary 1) and $G(y_1, \ldots, y_q)$ (in Corollary 2).

Remark. In the case that $Q \subset G \subset \mathbb{C}$, and we are interested in evaluating the polynomial (or the product of the matrix by vector) at some points in $\mathbb{C}^{n+2}$ (or $\mathbb{C}^{p+q+q}$), then the results hold even if we allow forming meromorphic functions of the y_i's in the algorithm, since we could have taken F to be the field of meromorphic functions (in one or q variables).

Remark. Let $Q \subset G \subset \mathbb{C}$, and instead of trying to compute $\Phi\underline{x} + \varphi$, we shall compute functions $\psi_1, \psi_2, \ldots, \psi_t$ such that $\| \psi_i - \sum_{j=1}^{n} \Phi_{i,j} x_j - \varphi_i \|_\infty \leq a$ for some $0 \leq a < \infty$. The p.oof of Theorem 1 is easily modified to show at least u m/d are required to compute the t function $\psi_1, \ldots, \psi_t$.

4. PRECONDITIONING OF COEFFICIENTS

The remarks at the end of the last section show that the number of m/d required to compute $\sum_{i=0}^{n} x_i y^i$ or $X\underline{y}$ can not be reduced by operations on the "y" variables. The following example, due to Motzkin, shows that by performing certain operations on the "x" variables, without counting those operations, will reduce the number of m/d .

Consider the polynomial $x_4 y^4 + x_3 y^3 + x_2 y^2 + x_1 y + x_0$. If we define $\lambda_1 = \frac{x_3 - x_4}{2x_4}$, $\lambda_2 = \frac{x_1}{x_4} - \lambda_1(\frac{x_2}{x_4} - \lambda_1(\lambda_1 + 1))$, $\lambda_3 = \frac{x_2}{x_4} - \lambda_1(\lambda_1 + 1) - \lambda_2$, $\lambda_4 = \frac{x_0}{x_4} - \lambda_2 \cdot \lambda_3$, we obtain

$$x_4 y^4 + x_3 y^3 + x_2 y^2 + x_1 y + x_0 = x_4[((y+\lambda_1)y + \lambda_2)y + y + \lambda_3) + \lambda_4] .$$

So if we do not count the m/d required to compute λ_1, λ_2, λ_3, and λ_4, this polynomial can be evaluated in 3 m/d only.

The assumption that operations on the "x" variables are not counted is quite reasonable when the same polynomial has to be evaluated at many points y . Since the λ_i's have to be evaluated only once, we see that the number of m/d per polynomial evaluation approaches 3 as the number of evaluations increases.

Motzkin [3] showed that under the above assumption at least $\lceil \frac{n+1}{2} \rceil$ m/d are necessary to evaluate $\sum_{i=0}^{n} x_i y^i$ (we use $\lceil x \rceil$ to denote the smallest integer $\geq x$), and showed that this bound can be reached. Motzkin s algorithm did not preclude the possibility that even though the x_i's are real numbers, the λ_i's may be complex. Pan [2] modified Motzkin's algorithms so that if all the x_i's are real, then the λ_i's are real, too.

In this section we will investigate the number of m/d required to compute the function studied in the last section under the assumption that operation on the "x" variables are not counted.

Let G be a field, $Q \subset G \subset \mathbb{C}$; let $F_{x_1, x_2, \ldots, x_n}$ be a field which includes $G(x_1, \ldots, x_n)$, and which can be imbedded in the field of continuous (except at isolated points) functions $f(x_1, \ldots, x_n)$ into $\mathbb{C}$ (in some domain), which vanish at isolated points. Similarly, let $F_{y_1, \ldots, y_m}$ be a field which includes $G(y_1, \ldots, y_m)$ and which is included in the field of continuous (except at isolated points) functions $f(y_i, \ldots, y_m)$ into $\mathbb{C}$, which vanish at isolated points.

In this section we will investigate algorithms over $(\mathcal{A},\mathcal{B})$ where $\mathcal{A}$ is the field $F_{x_1,\ldots,x_n} \cdot F_{y_1,\ldots,y_m}$ and $\mathcal{B} = F_{x_1,\ldots,x_n} \cup F_{y_1,\ldots,y_m}$.

Theorem 2. Let α be an algorithm (over $(\mathcal{A},\mathcal{B})$ as described above) computing $\Phi\underline{x} + \varphi$ where $\Phi_{i,j} \in G(y_1,\ldots,y_m)$ $i = 1,2,\ldots,t$ $j = 1,2,\ldots,n$. If there are u vectors in $\{\Phi_1,\ldots,\Phi_n\}$ such that no nontrivial linear combination of them (over G) is in G^t, then α has at least $\lceil \frac{u}{2} \rceil$ m/d.

Moreover, the conclusion holds even if multiplication or division by $g \in G$ are not counted.

Proof. Let $k_1 < k_2 < \ldots < k_0$ be the steps in α where m/d which are counted are performed. It is easily verified that $e_\alpha(k_1) = (\varphi_1 + \lambda_1) \times: (\varphi_2 + \lambda_2)$ (we use $\times:$ to denote a multiplication or division), where $\varphi_1, \varphi_2 \in F_{y_1,\ldots,y_m}$ and $\lambda_1, \lambda_2 \in F_{x_1,\ldots,x_n}$. Similarly,

$$e_\alpha(k_j) = \Big(\sum_{i=1}^{j-1} g_{i,j}\, e_\alpha(k_i) + \varphi_{2j-1} + \lambda_{2j-1}\Big) \times: \Big(\sum_{i=1}^{j-1} g'_{i,j}\, e_\alpha(k_i) + \varphi_{2j} + \lambda_{2j}\Big)$$

$j = 1,2,\ldots,s$, where $g_{i,j}, g'_{i,j} \in G$; $\varphi_{2j-1}, \varphi_{2j} \in F_{y_1,\ldots,y_m}$; and $\lambda_{2j-1}, \lambda_{2j} \in F_{x_1,\ldots,x_n}$. Also, $\sum_{j=1}^{n} \Phi_{i,j} x_j + \varphi_i = \sum_{j=1}^{s} g_{i,j}\, e_\alpha(k_j) + \varphi'_i + \mu_i$ $i = 1,2,\ldots,t$, where $g_{i,j} \in G$; $\varphi'_i \in F_{y_1,\ldots,y_m}$, $\mu_i \in F_{x_1,\ldots,x_n}$.

There exists a subneighborhood $N \subset C^n$ such that the functions λ_i $i = 1,2,\ldots,2s$ as well as μ_i $i = 1,2,\ldots,t$ are continuous in N and such that substituting $(x_1^0,\ldots,x_n^0) \in N$ for $x_1,\ldots,x_n$, ℓ_j is total for the resulting algorithm γ. With no loss of generality, let $\Phi_1, \Phi_2,\ldots,\Phi_u$ the columns of Φ such that no nontrivial linear combination of them is in G^t, then there exists a neighborhood $N' \subseteq \mathbb{C}^u$ such that $N' \times \{(x_{u+1}^0,\ldots,x_n^0)\} \subseteq N$ for some point $(x_{n+1}^0,\ldots,x_n^0)$. Therefore, the function $\lambda_1, \lambda_2,\ldots,\lambda_{2s}$ are continuous map of $N' \subset \mathbb{C}^u \to \mathbb{C}^{2s}$. If $u > 2s$, then this function

cannot be $1:1$ and therefore, there are two distinct points $x^0 = (x_1^0,\ldots,x_u^0, x_{u+1}^0,\ldots,x_n^0)$ and $x' = (x_1^1,\ldots,x_u^1, x_{u+1}^0,\ldots,x_n^0)$ such that $\lambda_i(x^0) = \lambda_i(x^1)$ $i = 1,2,\ldots,2s$. Therefore, if we denote $e_\alpha(k_i)(x^0)$ the evaluation of $e_\alpha(k_i)$ at x^0, and similarly for $e_\alpha(k_i)(x^1)$, we obtain that $e_\alpha(k_i)(x^0) = e_\alpha(k_i)(x^1)$ $i = 1,2,\ldots,s$. Substituting this relation in the evaluation of $\sum_{j=1}^{n} \Phi_{i,j} x_j + \varphi_i = \sum_{j=1}^{s} g_{i,j}\, e_\alpha(k_j) + \varphi_i' + \mu_i$ at the points x^0 and x^1, we obtain that $\sum_{j=1}^{u} \Phi_{i,j}(x_j^0 - x_j^1) = \mu_i(x^1) - \mu_i(x^0)$, so there exists a nontrivial linear combination (over $\mathbb{C}$) of $\Phi_1,\ldots,\Phi_n$ which is in $\mathbb{C}^t$, but since $\Phi_{i,j} \in G(y_1,\ldots,y_m)$, it follows that there exists a nontrivial linear combination (over G) of $\{\Phi_1,\ldots,\Phi_n\}$ which is in G^t contrary to assumption 2. So the assumption that $u > 2s$ is untenable and, therefore $s \geq \lceil \frac{u}{2} \rceil$.

<u>Corollary 3</u> (Motzkin). Every algorithm for computing $\sum_{i=0}^{n} x_i y^i$ requires at least $\lceil \frac{n}{2} \rceil$ m/d which do not depend only on the coefficients (or only on the variable y).

<u>Corollary 4</u>. Every algorithm for computing $X\underline{y}$, where X is a $p \times q$ matrix and $\underline{y}$ is a q vector, requires at least $\lceil \frac{pq}{2} \rceil$ m/d which do not depend only on the entries of X (or only on the entries of $\underline{y}$).

The possibility of achieving the lower bound of Corollary 3 was discussed by Motzkin [3] and Pan [2]. (In our terminology, they needed $F_{x_1,\ldots,x_n}$ to be the algebraic closure of $G(x_1,\ldots,x_n)$). The following algorithm for the product of a matrix by vector shows the possibility of approaching the bound of Corollary 4.

Let $X = (x_{i,j})$ be a $p \times q$ matrix, and $\underline{y} = (y_j)$ be a q vector. Let $\ell = \lfloor \frac{q}{2} \rfloor$. (We use $\lfloor u \rfloor$ to denote the integer part of u.) Define

$$\xi_i = \sum_{u=1}^{\ell} x_{i,2u-1} \qquad i = 1\ 2,\ldots,p$$

$$\eta = \sum_{u=1}^{\ell} y_{2u-1} \cdot y_{2u}, \quad \text{then}$$

$$\sum_{j=1}^{q} x_{i,j}\, y_j = \begin{cases} \sum_{u=1}^{\ell} (x_{i,2u-1} + y_{2u})(x_{i,2u} + y_{2u-1}) - \xi_i - \eta & \text{if } q = 2\ell \\ \sum_{u=1}^{\ell} (x_{i,2u-1} + y_{wu})(x_{i,2u} + y_{2u-1}) - \xi_i - \eta + x_{i,q} y_q & \text{if } q = 2\ell+1 . \end{cases}$$

Thus, if we do not count multiplications involving only the $x_{i,j}$'s (computing the ξ_i's), this algorithm requires $p\lceil \frac{q}{2} \rceil + \lfloor \frac{q}{2} \rfloor$ multiplications.

5. APPLICATIONS

The notion of preconditioning of the coefficients was introduced by Motzkin for reducing the number of multiplications when the same polynomial has to be evaluated at many different points. Similarly, the result of Corollary 4 can be used when the same matrix has to be multiplied by many vectors. A commonly occurring calculation of the product of the same matrix by many vectors is that of matrix multiplication.

Let $X = (x_{i,j})$ be a $p \times q$ matrix and $Y = (y_{i,j})$, a $q \times s$ matrix. Using the algorithm described in the end of the last section, we obtain the following algorithm (here we also use $\ell = \lfloor \frac{q}{2} \rfloor$).

Compute

$$\xi_i = \sum_{u=1}^{\ell} x_{i,2u-1} \cdot x_{i,2u}$$

$$\eta_j = \sum_{u=1}^{\ell} y_{2u-1,j} \cdot y_{2u,j}$$

and then use the identity

$$\sum_{k=1}^{q} x_{i,k}\, y_{k,j} = \begin{cases} \sum_{u=1}^{\ell} (x_{i,2u-1} + y_{2u,j})(x_{i,2u} + y_{2u-1,j}) - \xi_i - \eta_j & \text{if } q = 2\ell \\ \sum_{u=1}^{\ell} (x_{i,2u-1} + y_{2u,j})(x_{i,2u} + y_{2u-1,j}) - \xi_i - \eta_j + x_{i,q} y_{q,j} & \text{if } q = 2\ell + 1 . \end{cases}$$

The total number of multiplications is $p \cdot s \lceil \frac{q}{2} \rceil + \lfloor \frac{q}{2} \rfloor (p + s)$. For the particular case that $p = q = s = n$, we obtain that matrix multiplication can be done in roughly $\frac{1}{2} n^3$ multiplications rather than the n^3 multiplications usually required.

The identity used for the matrix multiplication algorithm can be used whenever inner products of vector have to be computed. Consider the case of having to compute T inner products involving N q-dimensional vectors. For each vector $\underline{x}_i = (x_1^i, x_2^i, \ldots, x_q^i)$, we compute $\xi_i = \sum_{u=1}^{\ell} x_{2u-1} \cdot x_{2u}$ $(\ell = \lfloor \frac{q}{2} \rfloor)$ and then use the same identity as for matrix multiplication. The total number of multiplications is $N \lfloor \frac{q}{2} \rfloor + T \lceil \frac{q}{2} \rceil$ as compared with Tq.

As application of the above method for computing the inner product, consider inversion of an $n \times n$ matrix and solution of n linear equations. If $n = k \cdot \ell$, we partition the $n \times n$ matrix to $\ell \times \ell$ matrix whose entries are $k \times k$ matrices. Applying Gaussian elimination to this $\ell \times \ell$ matrix, we obtain that matrix inversion requires $\frac{n^3}{2} + \frac{3n^2}{2} + \frac{n}{2}(k^2 - k)$ m/d (as compared with n^3 m/d using regular Gaussian elimination), and solution of n linear equations requires $\frac{n^3}{6} + \frac{3n^2}{2} + \frac{n}{6}(5k^2 + 3k + 6) - \frac{1}{3}(2k^3 + 4k)$ m/d (compared with $\frac{1}{3}(n^3 + 3n^2 - n)$ m/d using regular Gaussian elimination). Letting $k = 2$, we obtain that an $n \times n$ matrix (for n even) can be inverted using $\frac{n^3}{2} + \frac{3n^2}{2} + n$ m/d, and n linear equations can be solved using $\frac{n^3}{6} + \frac{3}{2} n^2 + \frac{10n}{3} - 8$ m/d.

V. V. Klyuyev and N. I. Kokovkin-Shcherbak [6] have proved that every algorithm for solving n linear equations which uses only row operation has to have at least $\frac{1}{3}(n^3 + 3n^2 - n)$ m/d. (This is the number required by Gaussian elimination.) It has been conjectured that this result holds even if the restriction on row operations is removed. The above algorithm for solving linear equations shows that conjecture to be false.

It should be mentioned that while the above algorithms reduced the number of multiplications, they increased the number of additions. Moreover, the sum of all the arithmetic operations in the new algorithms is about the same as in the regular matrix multiplication and in Gaussian elimination. This leads to the conjecture that the total number of arithmetic operations is constant, and that therefore any reduction in the number of m/d has to be at the expense of increase in the number of additions.

The algorithm for multiplying two $n \times n$ matrices involved only addition, subtraction, and multiplication, and is, therefore, an algorithm over $R[x_{1,1},\dots,x_{n,n}, y_{1,1},\dots,y_{n,n}]$ for some ring R with identity; the way we counted the multiplication means that we used $\mathcal{B} = \mathcal{R} \cup \{x_{i,j} \mid 1 \leq i,j \leq n\} \cup \{y_{i,j} \mid 1 \leq y_{i,j} \leq n\}$. For even number n we then obtained an algorithm for multiplying two $n \times n$ matrices requiring $\frac{1}{2}n^3 + n^2$ multiplications.

Let $f(n)$ be the minimum number of multiplications required to multiply two $n \times n$ matrices, when multiplications by a fixed element of R are not counted. The remaining of the paper is devoted to proving the following theorem.

<u>Theorem 3.</u> Either for every n, $f(n) \geq \frac{1}{2}n^3$, or there exist two constants k and $a < 3$ such that for every n, $f(n) \leq kn^a$.

Note that by Corollary 2, $f(n) \geq n^2$.

To prove Theorem 3 we need two lemmas. Let $\mathcal{R}$ be a ring with identity. We shall use $\mathcal{R}[x_1,\dots,x_n]$ to denote the ring extension of R by the commuting indeterminates $x_1,\dots,x_n$, and $R\{x_1,\dots,x_n\}$ to denote the ring extension of $\mathcal{R}$ by the noncommuting indeterminates $x_1,\dots x_n$. The functions of the product of two matrices can be viewed as either belonging to $R[x_{1,1},\dots,x_{n,n}, y_{1,1},\dots,y_{n,n}]$ or to $\mathcal{R}\{x_{1,1},\dots,x_{n,n}, y_{1,1},\dots,y_{n,n}\}$ and we can study the minimum number of multiplications required in either case. (We will assume that for our algorithms $\mathcal{B} = \mathcal{R} \cup \{x_{1,1},\dots,x_{n,n}, y_{1,1},\dots,y_{n,n}\}$).
Let $f^*(n)$ be the minimum number of multiplications

required to compute two $n \times n$ matrices in the noncommuting case. (Again, we assume that multiplication by a fixed element of R is not counted.)

Lemma 1. If for some n_0, $f^*(n_0) = n_0^a$, then there exists a constant k (depending on n_0 and a) such that for every n, $f^*(n) \leq kn^a$.

Proof. We will first prove that for $n = n_0^i$, $f^*(n) \leq n^a$, by induction on i.

For $i = 1$, this is asserted by the statement of the lemma. Assume the result holds for i. Partition the $n_0^{i+1} \times n_0^{i+1}$ matrices into $n_0 \times n_0$ matrices whose entries are $n_0^i \times n_0^i$ matrices. Using the algorithm for $n_0 \times n_0$ matrices, which require n_0^a multiplications, we construct an algorithm for multiplying two $n_0^{i+1} \times n_0^{i+1}$ matrices which require n_0^a multiplications of $n_0^i \times n_0^i$ matrices. By induction hypothesis, each of these latter multiplications of matrices can be done using only n_0^{ai} multiplication. So the total number of multiplications is $n_0^{a(i+1)} = (n_0^{a+1})^a$.

Let $k = n_0^a$. For any n, let $n_0^{i-1} < n \leq n_0^i$. There exists an algorithm for multiplying two $n_0^i \times n_0^i$ matrices requiring $n_0^{ai} > kn^a$ multiplications. Substituting in this algorithm $x_{i,j} = y_{i,j} = 0$ if $i > n$ or $j > n$ we obtain an algorithm for computing two $n \times n$ matrices requiring at most kn^a multiplications, and consequently $f^*(n) \leq kn^a$.

Lemma 2. $f^*(n) \leq 2f(n)$.

Proof. For simplicity of notation, we will use a single index to denote the "x" indeterminates, and a single index to denote the "y" indeterminates. Thus, the results of matrix multiplication is either in $R[x_1,\dots,x_m, y_1,\dots,y_m]$ or $R\{x_1,\dots,x_m, y_1,\dots,y_m\}$ where $m = n^2$.

Let $r = r_0 + \sum_{i=1}^{m} r_i x_i + \sum_{i=1}^{m} r'_i y_i + \sum_{i=j} r'_{i,j} x_i x_j + \sum_{i=j} y'_{i,j} y_i y_j + \sum_{i,j} r_{i,j} x_i y_j + \dots$ be an element of $R[x_1,\dots,x_m, y_1,\dots,y_m]$. We define $L_0(r) = r_0$, $L_{1,x}(r) = \sum_{i=1}^{m} r_i x_i$, $L_{1,y}(r) = \sum_{i=1}^{m} r_i y_i$, $L_2(r) = \sum_{i,j} r_{i,j} x_i y_j$,

$L_3(r) = r - L_0(r) - L_{1,x}(r) - L_{1,y}(r) - L_2(r)$. For every $r_1, r_2 \in R[x_1,\ldots,x_n, y_1,\ldots,y_n]$, we have the identity $r_1 \cdot r_2 = (r_2 - L_0(r_1))(r_2 - L_0(r_2)) + L_0(r_1)\cdot r_2 + L_0(r_2)\cdot r_1 - L_0(r_1)\cdot L_0(r_2)$. Let α be an algorithm for computing the product of two $n \times n$ matrices in $t = f(n)$ multiplications. Let $s_1, s_2,\ldots, s_t$ be the steps of α where the multiplications occur. Then for every $i = 1, 2,\ldots, t$ we have $e_\alpha(s_i) = e_\alpha(u_i)\cdot e_\alpha(v_i)$ for some u_i, v_i. Using the above identity and the assumption that multiplications by a fixed element of R are not counted, we can modify α such that for every $i = 1, 2,\ldots, t$ $L_0(e_\alpha(u_i)) = L_0(e_\alpha(v_i)) = 0$.

For every k $e_\alpha(k) = \sum_{i=1}^{t} r_{i,k}\, e_\alpha(s_i) + \sum_{i=1}^{m} r_i x_i + \sum_{i=1}^{m} r_i' y_i + r_0$ for some $r_{i,k}, r_i, r_i', r_0 \in R$. In particular, this holds for every step k such that $e_\alpha(k)$ is an entry in the product of the two matrices. For these k, however, $L_0(e_\alpha(k)) = L_{1,x}(e_\alpha(k)) = L_{2,y}(e_\alpha(k)) = L_3(e_\alpha(k)) = 0$, and $e_\alpha(k) = L_2(e_\alpha(k))$. Applying these operations to the above expression for $e_\alpha(k)$ and noting that they are linear operations (if we view $R[x_1,\ldots,x_m, y_1,\ldots,y_m]$ as a module over R), we obtain that for the k which compute the product of the matrices $e_\alpha(k) = \sum_{i=1}^{t} r_{i,k}\, L_2(e_\alpha(s_i))$. But by the assumption that $L_0(e_\alpha(u_i)) = L_0(e_\alpha(v_i)) = 0$, we obtain that $L_2(e_\alpha(s_i)) = L_{1,x}(e_\alpha(u_i))\, L_{1,y}(e_\alpha(v_i)) + L_{1,x}(e_\alpha(v_i))\cdot L_{1,y}(e_\alpha(u_i))$. The function $L_{1,x}(e_\alpha(u_i))$, $L_{1,x}(e_\alpha(v_i))$, $L_{1,y}(e_\alpha(u_i))$, and $L_{1,y}(e_\alpha(v_i))$ can be formed without a multiplication by algorithms over $\mathcal{R}\{x_1,\ldots,x_m, y_1,\ldots,y_m\}$, and therefore, it requires $2t$ multiplications in an algorithm over $\mathcal{R}\{x_1,\ldots,x_m, y_1,\ldots,y_m\}$ to form $L_{1,x}(e_\alpha(u_i))\cdot L_{1,y}(e_\alpha(v_i))$ and $L_{1,x}(e_\alpha(v_i))\cdot L_{1,y}(e_\alpha(u_i))$ $i = 1, 2,\ldots, t$. And since every entry in the product matrix is a linear combination (over R) of these $2t$ terms, an algorithm over $\mathcal{R}\{x_1,\ldots,x_m, y_1,\ldots,y_m\}$ for forming matrix multiplication using $2t$ multiplications can be constructed and therefore $f^*(n) \leq 2t = 2f(n)$.

<u>Proof of Theorem 3</u>. Assume that for some n, $f(n_0) < \frac{1}{2} n_0^3$. Then by Lemma 2, $f^*(n_0) < n_0^3$ and therefore, $f^*(n_0) = n_0^a$ for some $a < 3$, and by Lemma 1 there exists k such that

$f^*(n) \leq kn^a$ and, therefore, $f(n) \leq kn^a$.

The author has learned of a scheme discovered by V. Strassen, multiplying two 2×2 matrices with entries in a noncommutative ring, which requires 7 multiplications. This shows that two $n \times n$ matrices can be multiplied using $kn^{\log_2 7}$ multiplications.

ACKNOWLEDGEMENT

The author wishes to thank M. O. Rabin for many stimulating conversations. In particular, the suggestion to study approximate computations is his.

REFERENCES

[1] A. M. Ostrowski, "On two problems in abstract algebra connected with Horner's rule," Studies presented to R. vonMises, Academic Press, New York (1954) 40-48.

[2] V. Ya. Pan, "Methods of computing values of polynomials," Russian Mathematical Surveys, Vol. 21, No. 1 (January-February 1966) 105-136.

[3] T. S. Motzkin, "Evaluation of polynomials and evaluation of rational functions," Bull. Amer. Math. Soc., Vol. 61 (1955) 163.

[4] S. Winograd, "On the number of multiplications required to compute certain functions," Proc. National Acad. of Sci., Vol. 58, No. 5 (November 1967).

[5] S. Winograd, "A new algorithm for inner product," IEEE Trans. on Computers, Col. 17, No. 7 (July 1968) 693-694.

[6] V. V. Klyuyev and N. I. Kokovkin-Shcherbak, "On the minimization of the number of arithmetic operations for the solution of linear algebraic systems of equations," Stanford University Technical Report CS24, June 1965.